Factors affecting estimation of selenium automation

The various factors which effect and which you should consider for estimation of “Selenium” specific project are explained below:

#1 Scope of the project

Scope typically means identifying the correct test cases for automation. Apply “Divide & rule” strategy to accomplish it. Break your application in small chunks or modules and analyze each of them to come up with the appropriate test cases for automation.

The steps involved are:

1. Identify the various factors which will form the basis of identifying the candidate test cases.
2. Break the application into smaller modules
3. Analyze each module to identify the candidate test cases
4. Calculate ROI

For more details of how to identify the correct test case, please see my previous paper: Selection of correct test cases for Automation

#2 Complexity of the application

Steps involved here are:

1. Determine the Size the application based on the number of test cases that need to be automated.
2. Size complexity through Fibonacci series.
3. Identify the verification point and checkpoint of each test case

Here we have to establish the definition of big/medium and small sized application. This definition differs from an individual/group perspective. How you classify your application, depends can also be dependent upon the number of test cases.

For example:

If your application has 300 – 500 test cases to automate, you can consider it as the small-sized application. If the test cases are over 1500, it can be classified as complex. This factor can be different for the different application. For some, 1500 test cases to automate can be considered as small / medium scaled. So once you have identified the exact number of test cases, scale it to small/medium or large. Your strategy towards estimating the effort will hugely dependent on these criteria.

You have to also consider the different checkpoints and verification points for your test case. A test case can have more than 1 checkpoint

but will have only 1 verification point. In case you have more than 1 verification point, it is recommended to bifurcate into separate test cases. This will also ease your maintenance and enhancement of your test suite.

#3 Use of supporting tools/technologies

Steps involved here are:

1. Identify the framework and automation needs
2. Based on the needs, analyze and identify the tools to be used.
3. Identify the dependencies/implications of using the tool.

Selenium alone is not sufficient to build a framework or complete the automation. Selenium (Web driver) will only script the test case, but there are other tasks as well, like reporting the result, tracking the logs, taking screenshots etc.

To achieve these you need separate tools that will be integrated with your framework. So it is important here to identify these supporting entities which will best suit your requirement and will help to get a positive ROI

#4 Implementing the Framework

Here comes the tricky part J the steps involved are!!

1. Identify the input (pattern in which data is fed into script) and output (reports / test results) of your automation suite.

2. Design your input files. This may range from a simple text file to complex excel file. It is basically the file which will have your test data.
3. Design the folder structure based on your input parameters and
4. Implement the reporting feature (either in some excel file or using any tool like ReportNG)
5. Determine/implement logger in your framework
6. Implement the build tool in your framework
7. Implement the unit test framework (Junit or TestNG)

There are many other requirements apart from just scripting in test automation with Selenium, like reading the data from a file, reporting / tracking the test results, tracking logs, trigger the scripts based on the input conditions and environment etc. So we need a structure that will take care of all these scripts. This structure is nothing but your Framework.

Web applications are complex by nature because it involves lots of supporting tools and technology to implement. In a similar way, implementing the framework in Selenium is also tricky (I will not say complex) as it involves other tools to integrate. Since we know Selenium is NOT a tool but actually a collection/group of jar files, it is configured and not “Installed”, Selenium itself is not strong enough to build a complex framework. It requires a list of third-party tools for building a framework.

We have to remember here that there is nothing “Ready-made” in Selenium. For everything, we have to code, so provisions in estimation should be there for googling the errors and troubleshooting.

Here we have to understand that that Framework building is the most important aspect of your Automation effort. If your framework is rock solid, maintenance and enhancement become easier especially in the era of Agile, if your framework is good, you can integrate your tests in all the sprints easily.

I won’t be wrong if I say that this particular factor of designing the Framework should be the most important aspect of estimation. If needed (like in complex application) this factor should be again broken down into a separate WBS and estimation should be done.

#5 Learning & Training

Learning Selenium is a bit different than learning any other automation tool. It basically involves learning a programming language than just a scripting language (though script language helps while building a framework like you want to write a script that would invoke your automated scripts after doing the environment setting changes).

#6 Environment setup

Environment set up deals with (not limited to):-

- Setting up the code in the test environment
- Setting up code in production environment
- Writing scripts for triggering the automated tests.
- Developing the logic for reporting
- Establishing the frequency of running the scripts and developing logic for its implementation
- Creating text/excel files for entering the test data and test cases
- Creating property files for tracking the environments and credentials

#7 Coding/scripting and review

Before you actually start writing your tests, there are 2 prerequisites:

1. Candidate test cases should be handy
2. Framework is ready

Identify different actions that your web application does. It can be simple actions like navigation, clicking, entering text; or a complex action like connect to a database, handle flash or Ajax. Take one test case at a time, and identify what all action that particular test case does and estimate hours accordingly for per test case. The sum of all the hours for the entire test suite will give you the exact number.

Provision should be there for Review as well. The reviews are simply the code review which can be done either by a peer or a developer. Pair programming is the best option which is quick, but if it is not possible, based on the available resources or organizations review strategy, hours should be allocated for it.